Optimized Use of Memory to Increase Efficiency and Security in Cloud Computing

[1]Asir J. Tamboli, [2]Suhas B. Dombale, [3]Jayant C. Teware, [4]Prof. P. S. Sadaphule
[1]Student, [2]Student, [3]Student, [4]Professor
[1]Department of computer engineering,
[1]All India Shri Shivaji Memorial's Society Institute of Information Technology, Pune, India

Abstract : Cloud computing allows on-demand network access to a shared pool of configurable computing resources such as servers, storage and applications. These shared resources can be quickly provisioned to the consumers on the basis of paying only for whatever their usage. Cloud storage refers to the transfer of storage resources to the consumers over the Internet. Private cloud storage is restricted to a particular organization and data security risks are less compared to the public cloud storage. Hence, private cloud storage is built by exploiting the commodity machines within the organization and the important data is stored in it. When the utilization of such private cloud storage increases, there will be an increase in the storage demand. It leads to the expansion of the cloud storage with additional storage nodes. During such expansion, storage nodes in the cloud storage need to be balanced in terms of load. In order to maintain the load across several storage nodes, the data need to be migrated across the storage nodes. This data migration consumes more network bandwidth. The key idea behind this paper is to develop a dynamic load balancing algorithm based on deduplication to balance the load across the storage nodes during the expansion of private cloud storage

IndexTerms **- cloud, deduplication, security.**

I. INTRODUCTION

Now days purchasing and shopping at big malls is becoming a daily activity in metro cities. We can see huge rush at malls on holidays and weekends. The rush is even more when there are special offers and discount. People purchase different items Cloud computing enables on-demand network access to a shared pool of configurable computing resources such as servers, storage and applications. These shared resources can be rapidly provisioned to the consumers on the basis of paying only for whatever they use. Cloud storage refers to the delivery of storage resources to the consumers over the Internet. Private cloud storage is restricted to a particular organization and data security risks are less compared to the public cloud storage. Hence, private cloud storage is built by exploiting the commodity machines within the organization and the important data is stored in it. When the utilization of such private cloud storage increases, there will be an increase in the storage demand. It leads to the expansion of the cloud storage with additional storage nodes. During such expansion, storage nodes in the cloud storage need to be balanced in terms of load. In order to maintain the load across several storage nodes, the data need to be migrated across the storage nodes. This data migration consumes more network bandwidth. The key idea behind this Application is to develop a dynamic load balancing algorithm based on deduplication to balance the load across the storage nodes during the expansion of private cloud storage.

.

II. MOTIVATION

Main motivation of the system is to remove a load on cloud base servers and avoiding data Duplication using the some methodologies and algorithm. This system is basically perform on Hash Code detection techniques which is used for avoiding multiple storage of the files on the Cloud Server.

For the load balancing techniques system split the file into three chunks and stored into the three different location and the access is only for the valid person's or authorized persons only who has login credentials with the valid user key which is given by the admin.

III.LITERATURE SURVEY

i. **Zheng Yan, Senior Member, IEEE, Lifang Zhang, Wenxiu Ding, and Qinghai Zheng, Member, IEEE, " Heterogeneous Data Storage Management with Deduplication in Cloud Computing", 2017:** we propose a holistic and heterogeneous data storage management scheme in order to solve the problems like to make cloud data access control adapt to various scenarios and satisfy different user demands, duplicated data could be stored at the cloud in an encrypted form by the same or different users, in the same or different CSPs. The proposed scheme of this paper further realizes flexible cloud storage management with both data deduplication and access control that can be operated by either the data owner or a trusted third party or both or none of them and it can satisfy miscellaneous data security demands and at the same time save storage spaces with deduplication across multiple CSPs. This is a generic scheme to realize encrypted cloud data deduplication with access control, which supports the cooperation between multiple CSPs.

ii. **Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in Proc. 2012 IEEE INFOCOM, pp. 2581-2585, and 2012:** In this paper, they address two fundamental issues in a cloud environment: privacy and efficiency. They first review a private keyword-based file

retrieval scheme proposed by Ostrovsky et.al. Then, based on an aggregation and distribution layer (ADL), they present a scheme, termed efficient information retrieval for ranked query (EIRQ), to further reduce querying costs incurred in the cloud. Queries are classified into multiple ranks, where a higher ranked query can retrieve a higher percentage of matched files. Extensive evaluations have been conducted on an analytical model to examine the effectiveness of our scheme.

iii. **S. Kamara, and K. Lauter, "Cryptographic cloud storage," Financ. Crypto. Data Secur, pp. 136-149, Springer, 2010.** This paper is organized as follows. Section 1 describe, at a high level, a possible architecture for a cryptographic storage service. They consider both consumer and enterprise scenarios. They stress that this design is not intended to be a formal specification (indeed many important business and engineering questions would need to be addressed) but is only meant to serve as an illustration of how some of the new and non-standard cryptographic techniques that have been developed recently could be combined to achieve their goals. In Section 3 they give an overview of the benefits of a cryptographic storage service, e.g., reducing the legal exposure of both customers and cloud providers, and achieving regulatory compliance. In Section 4 they describe in more detail the relevant cryptographic techniques, including searchable encryption, proofs of storage and attribute-based encryption. Finally, in Section 5, they mention some cloud services that could be built on top of a cryptographic storage service such as secure back-ups, archival, health record systems, secure data exchange and ediscovery.

iv. **R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in Proc. 2009 ACM Workshop Cloud Compute. Secur. pp. 85-90, 2009:** In this paper, they characterize the problems and their impact on adoption. In addition, and equally importantly, they describe how the combination of existing research thrusts has the potential to alleviate many of the concerns impeding adoption. In particular, they argue that with continued research advances in trusted computing and computation-supporting encryption, life in the cloud can be advantageous from a business intelligence standpoint over the isolated alternative that is more common today.

v. **M. Kallahalla, E. Riedel, R. Swami Nathan, Q. Wang, and K. Fu, "Plutus: scalable secure file sharing on untrusted storage," in Proc. USENIX Conf. File Storage Technol., pp. 29–42, 2003.**: Plutus is a cryptographic storage system that enables secure file sharing without placing much trust on the file servers. In particular, it makes novel use of cryptographic primitives to protect and share files. Plutus features highly scalable key management while allowing individual users to retain direct control over who gets access to their files. They explain the mechanisms in Plutus to reduce the number of cryptographic keys exchanged between users by using file groups, distinguish file read and write access, handle user revocation efficiently, and allow an untrusted server to authorize file writes. They have built a prototype of Plutus on OpenAFS. Measurements of this prototype show that Plutus achieves strong security with overhead comparable to systems that encrypt all network traffic.

vi. **E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh, "SiRiUS: securing remote untrusted storage," in Proc. Netw. Distrib. Syst. Secur. Symp., pp. 131-145, 2003.**: This paper presents SiRiUS, a secure file system designed to be layered over insecure network and P2P filesystems such as NFS, CIFS, OceanStore, and Yahoo! Briefcase. SiRiUS assumes the network storage is untrusted and provides its own read-write cryptographic access control for file level sharing. Key management and revocation is simple with minimal out-of-band communication. File system freshness guarantees are supported by SiRiUS using hash tree constructions. SiRiUS contains a novel method of performing file random access in a cryptographic file system without the use of a block server. Extensions to SiRiUS include large scale group sharing using the NNL key revocation construction

IV. SYSTEM ARCHITECTURE

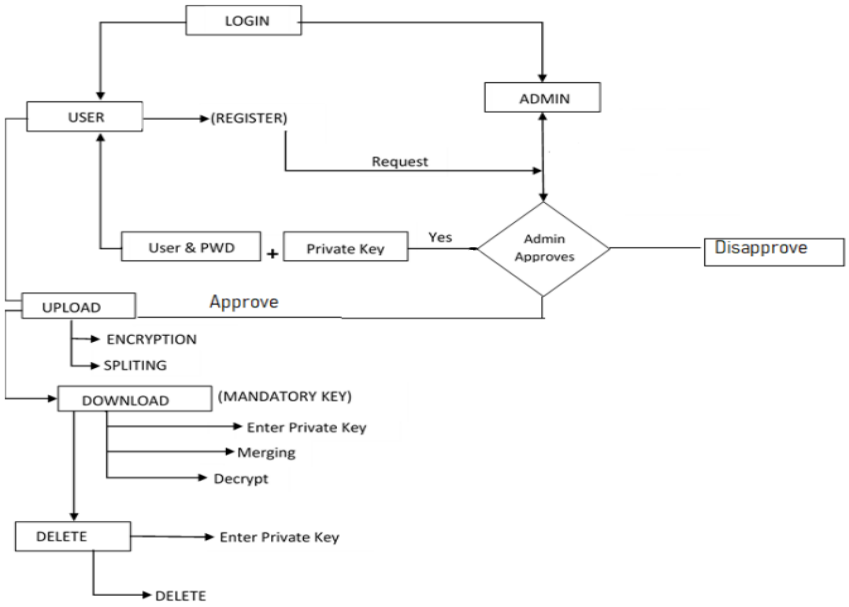

Fig.1 System Architecture

As shown in fig 3.1, This System has a functionality to ask information for the customer to the login and send the username, password and private key to the user with the help of the admin. Those have a login credentials as well as private key for the login who can easily perform upload, delete, and download operations. Using the Advanced Encryption standards (AES) and Message Digest (MD5) algorithm the data security and load balancing will be manage. The Hash Code is used to create code according to the file data and stored into the database if the code is same then Duplicate file message will be arrive otherwise the code is unique then file split into three different chunk and stored it into three Different location. After some quantum period of time files will be shuffled in directory.

V. ALGORITHM

i. Advanced Encryption standards (AES):

AES (acronym of Advanced Encryption Standard) is a symmetric encryption algorithm. The algorithm was developed by two Belgian cryptographer Joan Daemen and Vincent Rijmen. AES was designed to be efficient in both hardware and software, and supports a block length of 128 bits and key lengths of 128, 192, and 256 bits.

When you want to encrypt a confidential text into a decrypt able format, for example when you need to send sensitive data in e-mail.The decryption of the encrypted text it is possible only if you know the right password.

The schematic of AES structure is given in the following illustration −

- **Encryption Process**

Here, we limit to description of a typical round of AES encryption. Each round include of four sub-processes. The round process is depicted below −

1. Byte Substitution (SubBytes)

The 16 input bytes are substituted by viewing a fixed table (S-box) given in design. The outcome is in a matrix of four rows and four columns.

2. Shiftrows

Each of the four rows of the matrix is shifted to the left. Any items that fall off are re-inserted on the right side of row. Shift is carried out as follows −

a) First row is not shifted.
b) Second row is shifted one (byte) position to the left.
c) Third row is shifted two positions to the left.
d) Fourth row is shifted three positions to the left.

The outcome is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

3. MixColumns

Each column of four bytes is now converted using a special mathematical function. This function takes as input the four bytes of one column and outputs four totally new bytes, which replace the original column. The outcome is another new matrix consisting of 16 new bytes. It should be noted that this step is not implemented in the last round.

4. Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the outcomeing 128 bits are interpreted as 16 bytes and we begin another similar round.

- **Decryption Process**

The procedure of decryption of an AES ciphertext is similar to the encryption procedure in the reverse order. Each round consists of the four procedures conducted in the reverse order −

1. **Add round key**
2. **Mix columns**
3. **Shift rows**
4. **Byte substitution**

Since sub-procedures in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be independently implemented, although they are very closely related.

- **AES Analysis**

In current day cryptography, AES is widely accepted and supported in both hardware and software. Till date, no practical cryptanalytic attacks alongside AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

ii. MD5 Message-Digest Algorithm:

The MD5 message-digest algorithm is a commonly used hash function production a 128-bit hash value. While MD5 was primarily planned to be used as a cryptographic hash function, it has been found to suffer from all-encompassing weaknesses. It can silent be used as a checksum to prove data integrity, but only beside accidental fraud. It remains suitable for other non-cryptographic resolves, for example for shaping the screen for a particular key in a subdivided database

The trendy of MD5 is Ronald Rivest which is first printed in April 1992. Series started from MD2 then MD4, MD5, MD6.

1. **Append Padding Bits**

The message is "padded" (extended) so that its length (in bits) is same to 448, modulo 512. That is, the message is prolonged so that it is just 64 bits uneasy of actuality a multiple of 512 bits long. Padding is constantly achieved, even if the length of the message is even now identical to 448, modulo 512.

2. **Append Length**

A 64-bit picture of b (the length of the message earlier the padding bits were added) is appended to the result of the earlier step. In the questionable event that b is superior than 2^64, then only the low-order 64 bits of b are castoff. (These bits are appended as two 32-bit words and attached low-order word first in agreement with the previous resolutions.)

3. **Initialize MD Buffer**

A four-word buffer (A, B, C, and D) is used to calculate the message abstract. Here each of A, B, C, D is a 32-bit register.

4. **Process Message in 16-Word Blocks**

We first define four supplementary functions that each take as input three 32-bit words and produce as output one 32-bit word.

F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not (Z)
H(X,Y,Z) = X xor Y xor Z
I(X, Y, Z) = Y xor (X v not (Z))

In each bit situation F acts as a conditional: if X then Y else Z. The function F could have been well-defined by means of + instead of v since XY and not(X) Z will under no circumstances have 1's in the same bit position.) It is encouraging to note that if the bits of X, Y, and Z are autonomous and neutral, the each bit of F(X, Y, and Z) will be autonomous and neutral.

The functions G, H, and I are comparable to the function F, in that they act in "bitwise parallel" to produce their production from the bits of X, Y, and Z, in such a grace that if the identical bits of X, Y, and Z are autonomous and neutral, then respectively bit of G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be autonomous and neutral. Note that the function H is the bit-wise "xor" or "parity" meaning of its inputs**.**

5. **Output**

The message précis produced as output is A, B, C, D. That is, we initiate with the low-directive byte of A, and end with the high-directive byte of D.

iii. Round Robin Scheduling

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU. One of the most commonly used technique in CPU scheduling as a core. It is pre-emptive as processes are assigned CPU only for a fixed slice of time at most. The disadvantage of it is more overhead of context switching.

1. Each process is provided a fix time to execute, it is called a quantum.
2. Once a process is executed for a given time period, it is pre-empted and other process executes for a given time period.
3. Context switching is used to save states of pre-empted processes

VI. ADVANTAGES

i) Faster access and Effective
ii) Secure Data Via Private Key Encryption
iii) Avoid data loss while data Cloud server failure
iv) Maintain data Redundancy

VII. CONCLUSION

This system proposes the architecture of deduplication system for cloud storage environment and give the process of avoiding deduplication in each stage. In Client, system employ the file-level and chunk-level deduplication to avoid duplication. The algorithm also supports mutual inclusion and exclusion. Load sharing algorithm which is having policy to partitions the system into various domains and also having concept of cache manager and information dissemination for the various cloudlets.

References

[1] Zheng Yan, Senior Member, IEEE, Lifang Zhang, Wenxiu Ding, and Qinghua Zheng, Member, IEEE, " Heterogeneous Data Storage Management with Deduplication in Cloud Computing," 2017.

[2] Q. Liu, C. C. Tan, J. Wu, and G. Wang, "Efficient information retrieval for ranked queries in cost-effective cloud environments," in Proc. 2012 IEEE INFOCOM, pp. 2581-2585, 2012.

[3] S. Kamara, and K. Lauter, "Cryptographic cloud storage," Financ. Crypto. Data Secur, pp. 136-149, Springer, 2010.

[4] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in Proc. 2009 ACM Workshop Cloud Compute. Secur, pp. 85-90, 2009.

[5] M. Kallahalla, E. Riedel, R. Swami Nathan, Q. Wang, and K. Fu, "Plutus: scalable secure file sharing on untrusted storage," in Proc. USENIX Conf. File Storage Technol., pp. 29–42, 2003.

[6] E.-J. Goh, H. Shacham, N. Modadugu, and D.Boneh, "SiRiUS : securingremote untrusted storage," in Proc. Netw. Distrib. Syst. Secur. Symp., pp. 131-145, 2003